International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 11, Number 2, pages 463–491, July 2021

An External Definition of the One-Hot Constraint and Fast QUBO Generation for High-Performance Combinatorial Clustering

Masahito Kumagai
Graduate School of Information Sciences, Tohoku University
6-6-01, Aramaki Aza Aoba, Aoba-ku, Sendai, Miyagi, 980-8578, Japan

Kazuhiko Komatsu
Cyberscience Center, Tohoku University
6-3, Aramaki Aza Aoba, Aoba-ku, Sendai, Miyagi, 980-8578, Japan

Fumiyo Takano, Takuya Araki
Data Science Research Laboratories, NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki 211-8666, Japan

Masayuki Sato, Hiroaki Kobayashi
Graduate School of Information Sciences, Tohoku University
6-6-01, Aramaki Aza Aoba, Aoba-ku, Sendai, Miyagi, 980-8578, Japan

Received: February 15, 2021
Revised: May 5, 2021
Accepted: June 6, 2021
Communicated by Yasuaki Ito

Abstract

Recently, a clustering method using a combinatorial optimization problem, called *combinatorial clustering*, has been drawing attention due to the rapid spreads of quantum annealing and simulated annealing. Combinatorial clustering is performed by minimizing an objective function under a condition to satisfy a one-hot constraint. The objective function and the constraint function are generally formulated to a unified objective function of a QUBO (Quadratic Unconstrained Binary Optimization) problem using the method of the Lagrange multiplier. The coefficients of the QUBO function can be represented by a square matrix, which is called the QUBO matrix.

Although the Lagrange multiplier needs to be large enough to avoid violating the constraint, it is usually hard to be set appropriately due to the limitation of the bit precision. For example, the latest quantum annealer can handle values represented by only six or fewer bits. Even conventional computing systems cannot control the larger value of the Lagrange multiplier as the number of data points increases. Besides, the execution time for combinatorial clustering increases exponentially as the problem size increases. This is because the time for the QUBO matrix generation is long and a dominant factor of the total execution time when the problem size is large.

To solve these problems, this paper proposes combinatorial clustering that overcomes the limitation of the method of the Lagrange multiplier. The proposed method uses a QUBO solver that can externally define the one-hot constraint independent from the objective function, and the externally-defined constraint is satisfied by the operations with multiple bit-flips. As the QUBO function contains only the objective function, the method of the Lagrange multiplier is not necessary. The proposed method can optimize the objective function sufficiently even when the problem size is large. Since the constraint function is not included in the QUBO function, the proposed method also reduces the time for the QUBO matrix generation.

The experimental results obtained using the artificial and real data show that the proposed method can improve the quality of annealing-based clustering results in all data sets. The experimental results also clarify that the quality of the proposed method is almost equal to or better than that of quasi-optimal clustering methods such as K-means. The evaluation using the multiple traveling salesman problem shows that the proposed method can obtain shorter tour lengths than the conventional annealing-based clustering in all 13 cases and K-means++ in 6 out of 12 cases with a significant difference. Furthermore, the proposed method can accelerate the execution time for combinatorial clustering because there is no need to calculate the coefficients of the constraint function.

Keywords: Combinatorial Clustering, Simulated Annealing, Quantum Annealing, Constraint Function.

1 Introduction

Today, the demand for high-performance clustering in the field of data science is growing as it is widely used in the analysis of big data. For example, the industry uses the clustering of customer information to provide recommendations and appropriate support to customers. Therefore, clustering is an essential tool in modern social systems.

Clustering is performed by minimizing an objective function. The objective function for clustering evaluates the sum of intra-cluster and inter-cluster distances among data points [2]. The number of combinations of classification results increases as the number of data points increases. Therefore, it is difficult to search for a combination that minimizes an objective function in the case of large data sets.

In order to avoid searching a vast amount of combinations, K-means approximates an objective function to reduce the computational complexity by performing local search optimization [3]. However, the approximate objective function does not always provide an accurate clustering result because the clustering result is based on the local optimal solutions.

Recently, *combinatorial clustering* has been drawing attention. Combinatorial clustering solves clustering problems as *a combinatorial optimization problem.* Due to the advances in quantum annealing (QA) and simulated annealing (SA), finding the exact optimal solution that minimizes an objective function gradually becomes practical. Especially, QA takes advantage of quantum superposition effects to quickly solve combinatorial optimization problems. Kumar et al. [4] have indicated that, for combinatorial clustering, QA can quickly obtain the exact optimal solution that minimizes an objective function and achieves high-quality clustering results.

In combinatorial clustering using QA, a *binary variable* q_a^i is used to express whether a data point i belongs to a particular cluster a or not. Since a data point i belongs to only one cluster, only one of the binary variables $q_1^i \sim q_K^i$ becomes one. This constraint is called *a one-hot constraint.* Combinatorial clustering can be performed by minimizing an objective function under the condition to satisfy this constraint.

However, in the conventional method of combinatorial clustering, an objective function and a constraint function are formulated to a QUBO (Quadratic Unconstrained Binary Optimization) problem by the method of the Lagrange multiplier. While the Lagrange multiplier needs to be large enough to avoid violating the one-hot constraint, it is usually hard to be set appropriately due to the limitation of the bit precision. For example, the latest quantum annealer can handle values represented by only six or fewer bits [4,5]. Even in the case of conventional computing systems with

A short version of this work [1] has appeared in the proceedings of the CANDAR'20.

high bit precision, their bit precisions are not sufficient when the problem size is large. Moreover, the time for the QUBO matrix generation is one of the dominant computations in combinatorial clustering. This is because the execution time is long in the case where the QUBO matrix represents the one-hot constraint as coefficients.

To overcome the limitation of the method of the Lagrange multiplier and improve the quality of combinatorial clustering, this paper proposes a different approach to combinatorial clustering. The idea of the proposed method is to externally define the one-hot constraint independent from an objective function in a QUBO solver. In the solver, the externally-defined constraint is satisfied by the operations with multiple bit-flips. Furthermore, this paper also proposes a method to accelerate the generation of a QUBO matrix.

The rest of this paper is organized as follows. Section 2 describes clustering algorithm, and their quantum annealing and simulated annealing implementations. Section 3 proposes a combinatorial clustering method based on an externally-defined one-hot constraint. Section 4 evaluates the proposed method in terms of the quality and the execution time in clustering. Section 4 also discusses the performance of the proposed method by using a real application. Section 5 describes related work on the proposed method. Section 6 concludes this paper.

2 Combinatorial Clustering by Annealing Algorithms

2.1 Combinatorial Clustering

This subsection first describes conventional clustering methods and annealing algorithms. Second, the clustering based on the annealing algorithm is presented.

Combinatorial clustering minimizes the sum of intra-cluster distances among data points. The objective function is represented as the following equation.

$$H = \frac{1}{2}\sum_{a=1}^{K}\sum_{C(i)=C_a}\sum_{C(i')=C_a} d(x_i, x_{i'}). \tag{1}$$

Here, K is the number of clusters. x_i indicates a coordinate of data point i, and $d(x_i, x_{i'})$ indicates the distance between i and i'. $C(i) = C_a$ indicates that data point i belongs to cluster a. For clustering, the smaller the value H of the objective function is, the higher quality the clustering result is. Therefore, finding the combination of data that minimizes the objective function of combinatorial clustering guarantees that a global optimal solution is found [4].

However, finding the global optimal solution of combinatorial clustering becomes difficult as the problem size increases [6]. A problem search space of clustering, $S(N, K)$, that allocates N data points to K clusters is represented by the following equation [6].

$$S(N, K) = \frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\begin{pmatrix} N \\ i \end{pmatrix} i^N. \tag{2}$$

For example, $S(19, 4) \approx 1.1 \times 10^{10}$, and the number of combinations of the clustering results is very large [4].

There are two approaches to this problem. The first is to perform a quasi-optimal search, such as K-means [3]. The second method is to reduce computational complexity by using efficient search algorithms based on SA or QA [4, 7].

2.2 K-means Algorithm

K-means is a heuristic algorithm that efficiently solves the quasi-optimal solution of clustering [8]. In Eq. (1), the distances among all the data points need to be computed. Instead, in K-means, the distances between the centroid and each data point in the cluster are computed. Initially, the centroids of K clusters are randomly selected. Then, each data point is assigned to one of the clusters whose centroid is the nearest. Next, the centroids of the generated clusters are computed, and the data points are classified into the clusters in the same way. These computations are repeated until there is no change in the cluster, or the amount of change in the clustering result goes below a certain threshold [8, 9].

The K-means algorithm is mathematically approximated by the following equation [9].

$$SSE = \sum_{a=1}^{K} \sum_{C(i)=C_a} (x_i - \mu_{C_a})^2. \tag{3}$$

Here μ_{C_a} is a cluster centroid of C_a. Eq. (3) calculates the sum of squared errors of prediction (SSE) in clusters, which is also called *Inertia*.

However, since Eq. (3) is an approximation of the objective function shown in Eq. (1), the local optimal solution for clustering may be obtained [4]. Recently, accelerating the optimization of Eq. (1) using annealing algorithms has been discussed intensively to obtain the global optimal solution of clustering.

2.3 Annealing Algorithms using QUBO

Annealing algorithms such as SA and QA are expected to accelerate solving a combinatorial optimization problem. The combinatorial optimization problem is a problem to find the optimal solution from combinations of multiple variables. In the annealing algorithms, the combinatorial optimization problem is represented as a QUBO problem, which minimizes an objective function of a quadratic polynomial described by binary variables [10]. The optimal solution is found by minimizing the value of the objective function, called a *Hamiltonian*.

The Hamiltonian of QUBO is represented by Eq. (4).

$$H = \sum_{i<j} a_{i,j} q_i q_j + \sum_i b_i q_i^2. \tag{4}$$

Here, q_i represents a value of 0 or 1, called a *qubit*. When there are n qubits, the range of integer i and j is from 0 to $n-1$. The flipping of a binary variable from 0 to 1 or 1 to 0 is called a *bit-flip* operation. In Eq. (4), since all terms are a product of two qubits, they can be expressed as a product of a QUBO matrix and a vector, as shown in the following equations.

$$H = (q_0, q_1, \cdots, q_{n-1}) Q \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_{n-1} \end{pmatrix}, \tag{5}$$

$$Q = \begin{pmatrix} b_0 & a_{0,1} & \dots & a_{0,n-2} & a_{0,n-1} \\ 0 & b_1 & \dots & a_{1,n-2} & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & b_{n-2} & a_{n-2,n-1} \\ 0 & 0 & \dots & 0 & b_{n-1} \end{pmatrix}. \tag{6}$$

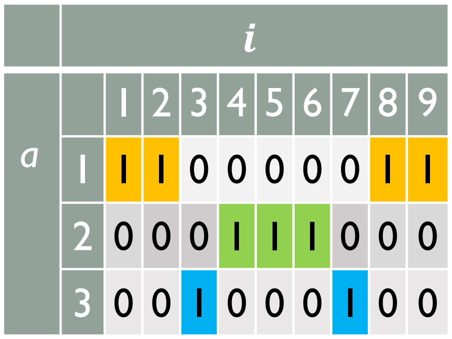
Here, Q indicates a QUBO matrix. Therefore, the combinatorial optimization problem can be represented in the form of a QUBO matrix.

To solve combinatorial clustering by annealing algorithms, Eq. (1) is transformed into a QUBO function by introducing a binary variable, as shown in the following equation.

$$H = \frac{1}{2} \sum_{i,j=1}^{N} d(x_i, x_j) \sum_{a=1}^{K} q_a^i q_a^j + \sum_{i=1}^{N} \lambda_i (\sum_{a=1}^{K} q_a^i - 1)^2. \tag{7}$$

The binary decision variable q_a^i is defined as the following equation.

$$q_a^i = \begin{cases} 1 & (C(i) = C_a) \\ 0 & (C(i) \neq C_a). \end{cases} \tag{8}$$



		i								
		1	2	3	4	5	6	7	8	9
a	1	1	1	0	0	0	0	0	1	1
	2	0	0	0	1	1	1	0	0	0
	3	0	0	1	0	0	0	1	0	0

Figure 1: An example of combinatorial clustering by one-hot encoding.

λ_i is defined as the following equation.

$$\lambda_i \geq (N - K)\tilde{d}. \tag{9}$$

$\tilde{d}$ is the pair-pointwise maximum of $d(x_i, x_j)$ for all i and j. The first term of Eq. (7) indicates the objective function obtained by transforming Eq. (1). The second term is the constraint function with one-hot constraint that is required by introducing binary variables. Figure 1 shows an example of the solution that satisfies the one-hot constraint. If the variable q_a^i is one, data point i belongs to cluster a. By minimizing the second term of Eq. (7), the one-hot constraint is satisfied. When data points N are clustered into K clusters, NK variables are required. All the data points need to be clustered into one of K clusters. Besides, the normalization of $d(x_i, x_j)$ such that $\tilde{d} = 1$ results in follwing equation from Eq. (9).

$$\lambda = (N - K). \tag{10}$$

In this paper, combinatorial clustering using one-hot encoding is called one-hot clustering.

To solve a combinatorial optimization problem by avoiding whole search of combinations, a heuristic method of SA is used [11]. Figure 2 shows the difference between a hill climbing algorithm [12] and a SA algorithm. As shown by the green arrow in Figure 2, the hill climbing algorithm calculates Hamiltonian $H1$ of a random initial combination $q_0 \sim q_{n-1}$. Then, another Hamiltonian $H2$ is calculated. Here, the only binary variable q_i in a combination of Hamiltonian $H1$ is flipped in a combination of Hamiltonian $H2$. By comparing Hamiltonians $H1$ with $H2$, the combination of the smaller Hamiltonian is adopted as the more suitable solution. Thereafter, Hamiltonian $H3$ is computed with only one bit-flip operation for the combination of $H2$ in the same way. If $H3$ is smaller than $H2$, the combination of $H3$ is adopted.

However, this simple procedure may cause a local optimal solution. To get out of a local optimal solution, SA also attempts a search for a larger Hamiltonian, as shown by the red arrows in Figure 2. SA adopts the combination obtained by Hamiltonian $H3$ as the more suitable solution even when Hamiltonian $H3$ of the combination is larger than Hamiltonian $H2$ of another combination with only one bit-flip operation. The condition to adopt larger Hamiltonian $H3$ to avoid a local optimal solution is expressed as the following equation using temperature variable T (> 0). The $rand[0, 1]$ function performs uniform random sampling within the real interval including 0 and 1.

$$\exp(-\frac{H2 - H1}{T}) > rand[0, 1]. \tag{11}$$

This bit-flip operation probabilistically prevents from the local optimal solution. In this case, temperature T decreases as time increases [11, 13]. Due to thermal fluctuations, a combination of a higher Hamiltonian tends to be adopted at the beginning of the search. At the end of the search, a combination of a lower Hamiltonian is adopted. As a result, SA can find the optimal solution. This process is repeated until the pre-defined number of iterations, called sweeps.

The other way to solve a combinatorial optimization problem is to use QA. QA uses quantum fluctuations instead of thermal fluctuations to search the solution space [14]. In QA, a quantum

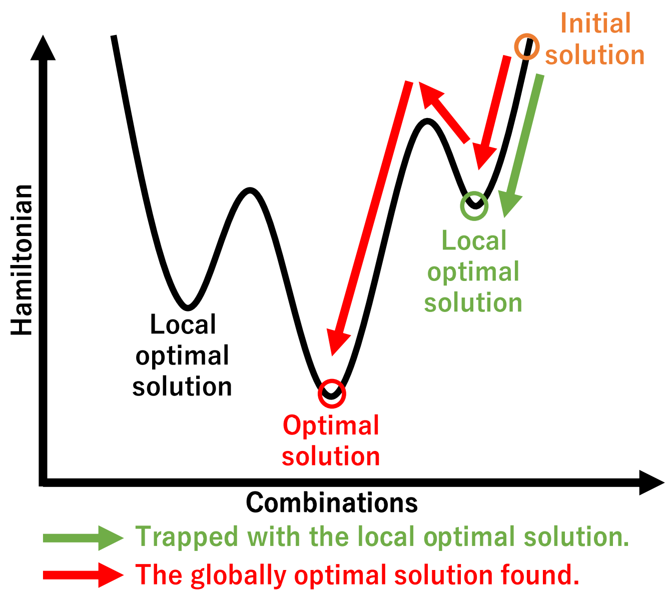


Figure 2: Difference between a hill climbing algorithm (green arrow) and a SA algorithm (red arrows).

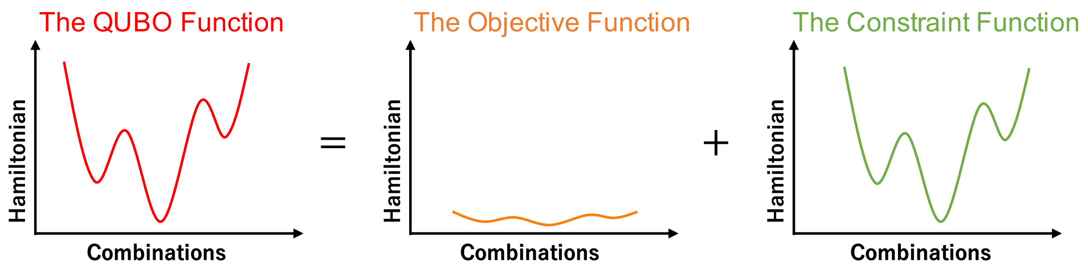


Figure 3: The conventional QUBO function with both an objective and a constraint function.

bit (qubit) is the smallest unit of information. One bit represents either state "0" or "1" while one qubit is a superposition of both states "0" and "1." A superposition of qubits converges to either state "0" or "1" only when it is observed. While SA needs a neighborhood search to invert one of the bits of the combination, QA can search for the global solution space by superposition states [15]. Besides, even in the case when trapped in a optimal local solution, QA can find the optimal solution by using the quantum tunneling effect that slips through the potential barrier of the local minimal value.

D-Wave Systems Inc. has developed D-Wave 2000Q as a dedicated quantum annealer for QA [16]. When solving a combinatorial optimization problem in D-Wave 2000Q, the objective function is formulated into the QUBO matrix. The QUBO matrix is sent to D-Wave 2000Q to solve the combinatorial optimization problem by hardware.

2.4 Problems of the annealing-based clustering algorithm

Figure 3 presents the conventional QUBO function with both the objective function and the constraint function in a search space. The coefficients of the constraint function is set to a large enough value due to the large Lagrange multiplier in Eq. (9). The search space for the solution of the constraint function has deep valleys, as shown in Figure 3. In contrast, the coefficients of the objective function are small compared to those of the constraint function. Figure 3 shows that the search space for the solution of the objective function is shallow compared to that of the constraint function. The influence of the objective function is small relative to the constraint function in the QUBO function. Since the QUBO function does not sufficiently represent the characteristics of the objective function, it is difficult to optimize.

Moreover, QA is optimized using the Ising spin glass in actual machines [17], where *spin variables* $\sigma \in \{-1,+1\}$ and binary variables $q \in \{0,1\}$ are transformed by $q = \frac{1}{2}(\sigma+1)$. To investigate the limitation of combinatorial clustering by a quantum annealer, the binary variables in Eq. (7) can be further transformed into the following equation by using spin variables [4].

$$H = \frac{1}{8}\sum_{i,j=1}^{N} d(x_i,x_j)\sum_{a=1}^{K}\sigma_a^i\sigma_a^j + \frac{1}{4}\lambda\sum_{i=1}^{N}\sum_{a\neq b}\sigma_a^i\sigma_b^i$$
$$+\frac{1}{4}\sum_{i,j=1}^{N} d(x_i,x_j)\sum_{a=1}^{K}\sigma_a^i + \frac{1}{2}\lambda(K-2)\sum_{i=1}^{N}\sum_{a=1}^{K}\sigma_a^i. \quad (12)$$

In Eq. (12), the first and third terms are obtained by transforming the first term in Eq. (7), and the second and fourth terms are obtained by transforming the second term in Eq. (7). Here, the constant term is ignored because its value does not change. Assuming a machine that can handle n-bit integer types, the maximum value of the coefficients is $2^{n-1}-1$ [18]. As the maximum coefficient in Eq. (12) is the fourth term, the following equation is obtained.

$$\frac{1}{2}\lambda(K-2) = 2^{n-1}-1. \quad (13)$$

Then, the maximum value of the Lagrange multiplier can be driven by the following equation.

$$\lambda = \frac{2(2^{n-1}-1)}{K-2}. \quad (14)$$

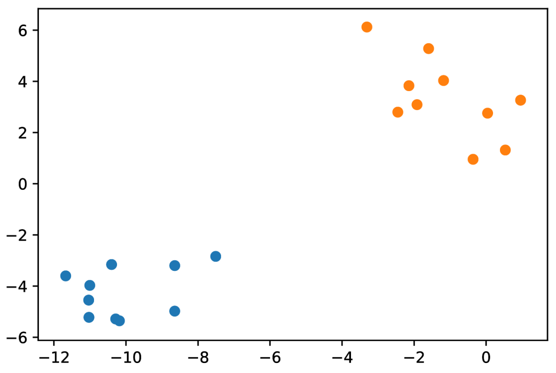
Since $\lambda \geq (N-K)d(x_i,x_j)$ from Eq. (9), the following formula expansion can be derived for the distances $d(x_i,x_j)$ when the coefficients are given to the Hamiltonian.

$$d(x_i,x_j) \quad \leq \quad \frac{2(2^{n-1}-1)}{(N-K)(K-2)}. \quad (15)$$

Current quantum annealers support only about 6-bit precision for coefficients at most [4,5]. For example, when the number of clusters K is 4, and the number of data points N is 3, $d \leq 62$. On the actual quantum annealer, $\frac{1}{8}d < 8$ from Eq. (12). This means that the coefficients have only about 3 bits of precision at most. As the number of data increases, the bit precision of the coefficients becomes smaller, resulting in difficulty in practical clustering.

Even in the case of conventional computers that can handle data types with a larger precision, the bit precision of combinatorial clustering becomes a severe problem. Assuming a computer that can use 32-bit integer types, $N = 1,500$ and $K = 6$, $\frac{1}{8}d$ should be less than $89,837$ [4]. This means that the precision is about 16 bits, around half the actual precision. As the number of data points and/or clusters increases, the denominator of Eq. (15) becomes further large. As a result, the precision requirements for d become severe. The hardware precision is strictly required by unifying an objective and a constraint function by the Lagrange multiplier.

Figure 4 shows the results of the one-hot clustering using SA when the number of clusters $K = 2$. The numbers of data points N in Figures 4 (a) and (b) are 20 and 200, respectively. From Figure 4, it is shown that combinatorial clustering does not work well when the number of data points is 200. This is because the 64-bit precision is not enough in the case of $N = 200$, while the 64-bit precision is enough in the case of $N = 20$. The use of the Lagrange multiplier causes the bit precision problem. Therefore, it is necessary to satisfy the constraint apart from the minimization of the objective function.


(a) The number of data points N is 20.

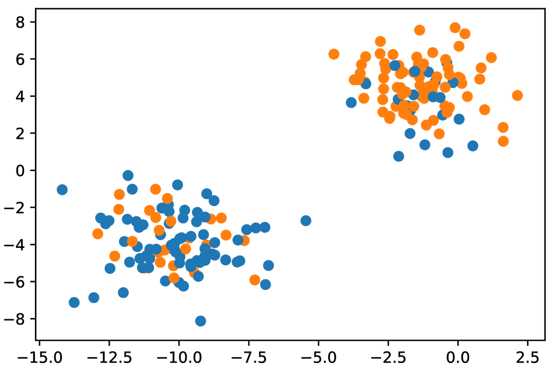

(b) The number of data points N is 200.

Figure 4: One-hot clustering using SA at $K = 2$.

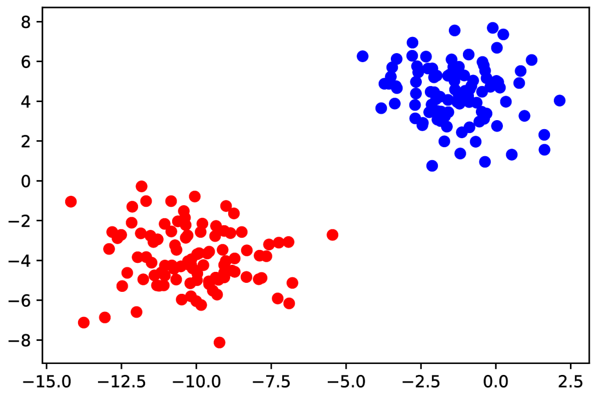


Figure 5: Binary clustering using SA at $K = 2$ and $N = 200$.

One of the methods to solve the bit precision problem is binary clustering [4]. Binary clustering is shown as the following equation.

$$H = \frac{1}{2} \sum_{i,j=1}^{N} d(x_i, x_j) s_i s_j. \tag{16}$$

Here s_i is defined as the following equation.

$$s_i = \begin{cases} +1 & (C(i) = C_0) \\ -1 & (C(i) = C_1). \end{cases} \tag{17}$$

Since binary clustering does not use a constraint function, it relaxes the bit precision problem. Figure 5 shows that binary clustering can be performed even with the problem size that the one-hot clustering failed. However, the number of clusters is limited to two, which is not practical at all. Thus, clustering that can handle any number of clusters is strongly required.

3 Combinatorial Clustering Based on an Externally-Defined One-Hot Constraint

The QUBO function of combinatorial clustering uses the Lagrange multiplier method to combine an objective function and a constraint function into a QUBO matrix. The sufficient precision cannot

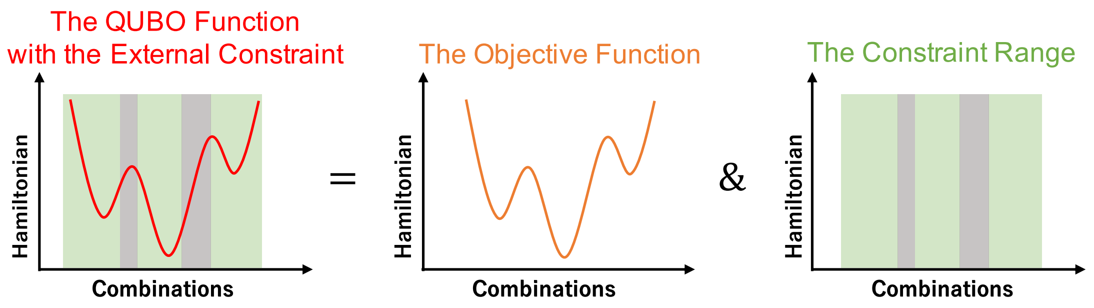


Figure 6: The proposed QUBO function with an external constraint.

Algorithm 1 A multiple bit-flips algorithm in simulated annealing with an externally-defined one-hot constraint.

```
1: Select a variable $q_a^i$ to flip
2: Get variables $q_1^i \sim q_K^i$ related with the constraint of $q_a^i$
3: if The number of 1 in the variables $q_1^i \sim q_K^i$ is 1 then
4:     if $q_a^i$ is 1 then
5:         Flip another variable from $q_1^i \sim q_K^i$ except $q_a^i$
6:     else
7:         Flip a variable of its value equals to 1 in $q_1^i \sim q_K^i$
8:     end if
9: end if
10: Flip a variable $q_a^i$
```

be given to coefficients of the QUBO matrix because the bit precision is limited in the quantum annealer. The Lagrange multiplier is needed when the QUBO matrix includes both the objective function and the constraint function. Therefore, this paper proposes combinatorial clustering that overcomes the limitation due to the Lagrange multiplier.

Moreover, the generation of the QUBO matrix by the conventional method dominates the execution time in combinatorial clustering. Therefore, this paper also proposes an efficient method to generate the QUBO matrix for the acceleration of combinatorial clustering.

3.1 Combinatorial Clustering using Simulated Annealing with an External Constraint

The proposed method uses a QUBO solver that can externally define the one-hot constraint independent from the objective function [19]. In the QUBO solver, a constraint is satisfied by the algorithm of *SAEC (Simulated Annealing with an External Constraint)*. Figure 6 presents the proposed QUBO function with the external constraint in a search space. The green area in the figure is the region that satisfy the constraints. SAEC receives the externally-defined constraint among variables as input data. There is no need to include the coefficients of the constraint function in the QUBO matrix when SAEC is used. Therefore, the QUBO function sufficiently represents the objective function as shown in Figure 6. In SAEC, the solution is checked if it satisfies the constraint after each sweep. If the flip of one of the variables does not satisfy the constraint, it refers to the constraint information and flips another variable to satisfy the constraint.

In Figure 1, only one variable among q_1^i, q_2^i, and q_3^i is 1, and the others are 0. Variable $q_1^1 = 1$ means data point $i = 1$ belongs to cluster $a = 1$. If a variable $q_1^1 = 1$ is flipped, all variables q_1^1, q_2^1, and q_3^1 become zero, which violates a constraint. Data point $i = 1$ does not belong to any cluster. Algorithm 1 shows how to flip variables by SAEC. To prevent the constraint violation, SAEC controls the solution to satisfy the constraint by simultaneously flipping either q_2^i or q_3^i when q_1^i is flipped. This allows SAEC to efficiently search the solution space while satisfying the

Listing 1: Generating a QUBO matrix of the conventional method using PyQUBO.

```
from pyqubo import Array, Constraint
from .utils import min_max

def get_pyqubo(dist, n_clusters):
  leng = len(dist)
  lam = leng - n_clusters
  dist = min_max(dist)

  qbit = Array.create('qbit',
    shape=(leng, n_clusters),
    vartype='BINARY')
  H = 0.5 * sum(dist[i,j] * qbit[i,a] * qbit[j,a]
    for i in range(leng)
    for j in range(leng)
    for a in range(n_clusters))
  + lam * Constraint(sum((sum(qbit[i,a]
    for a in range(n_clusters)) - 1) ** 2
    for i in range(leng)), label='ONE_HOT')
  model = H.compile()
  qubo, offset = model.to_qubo()

  return qubo
```

constraint. The search for a solution to reduce the value of the objective function is performed by the conventional SA algorithm. The operation with multiple bit-flips of Algorithm 1 are performed so that the constraints are satisfied.

Combinatorial clustering using annealing algorithms needs to generate a QUBO matrix that represents a QUBO problem to be optimized by SAEC. Listing 1 shows the implementation of the generation of the QUBO matrix from the Hamiltonian in combinatorial clustering of Eq. (7) by using PyQUBO. As it is difficult to generate the QUBO matrix from the Hamiltonian, PyQUBO has been developed to save the effort of generating the QUBO matrix by programmers [20]. In Listing 1, `get_pyqubo` receives `dist`, the distance matrix of data, and `n_clusters`, the number of clusters, and calculates the QUBO matrix. `min_max` is the normalization function for the given distance matrix. The Lagrange multiplier is adopted as Eq. (10).

There are two concerns about the long execution time for PyQUBO. The one is an algorithmic problem. PyQUBO automatically calculates coefficients of the QUBO matrix based on the Hamiltonian. In the process of generating the QUBO matrix, PyQUBO expands the formula of the Hamiltonian, and adds up all the QUBO coefficients of the similar terms. There is a regularity to generate coefficients of a QUBO matrix from the Hamiltonian of combinatorial clustering. Therefore, the QUBO coefficients can be calculated in advance using regularity.

The other is an implementation problem. PyQUBO uses `append` to add coefficients to the QUBO matrix in the Python dictionary format. A Python dictionary is a data structure that is commonly known as an associative array. `append` creates a new data structure by adding elements to an existing data structure. However, `append` is inefficient because it allocates a new memory space that replicates the existing memory space and extends a new element. Therefore, the generation of QUBO matrices can be accelerated by reserving the necessary amount of memory in advance.

3.2 Compact QUBO Definition for Combinatorial Clustering

This subsection describes a method to generate the QUBO matrix. This method effectively assigns QUBO coefficients to the QUBO matrix in order to accelerate the generation of the QUBO matrix. Among the Hamiltonians of combinatorial clustering in Eq. (7), the objective function and the constraint function are given by the following equations.

$$\begin{aligned} H_{objective} &= \frac{1}{2}\sum_{i,j=1}^{N} d(x_i,x_j)\sum_{a=1}^{K} q_a^i q_a^j \\ &= \sum_{\substack{i=1\\ j>i}}^{N} d(x_i,x_j)\sum_{a=1}^{K} q_a^i q_a^j \qquad (18) \end{aligned}$$

and

$$H_{constraint} = \sum_{i=1}^{N} \lambda_i H_{one-hot}. \qquad (19)$$

Here, $H_{one-hot}$ is defined by Eq. (20). Moreover, Eq. (20) is transformed into Eq. (21) as follows.

$$\begin{aligned} H_{one-hot} &= (\sum_{a=1}^{K} q_a^i - 1)^2 \qquad (20) \\ &= (\sum_{a=1}^{K} q_a^i)^2 - 2\sum_{a=1}^{K} q_a^i + 1 \\ &= \sum_{a=1}^{K}\sum_{b=1}^{K} q_a^i q_b^i - 2\sum_{a=1}^{K}(q_a^i)^2 + 1 (\because q_a^i = (q_a^i)^2) \\ &= \sum_{\substack{a,b=1\\ a=b}}^{K} q_a^i q_b^i + \sum_{\substack{a,b=1\\ a\neq b}}^{K} q_a^i q_b^i - 2\sum_{a=1}^{K}(q_a^i)^2 + 1 \\ &= \sum_{a=1}^{K}(q_a^i)^2 + 2\sum_{\substack{a=1\\ b>a}}^{K} q_a^i q_b^i - 2\sum_{a=1}^{K}(q_a^i)^2 + 1 \\ &= -\sum_{a=1}^{K}(q_a^i)^2 + 2\sum_{\substack{a=1\\ b>a}}^{K} q_a^i q_b^i + 1. \qquad (21) \end{aligned}$$

The fact that $q_a^i = (q_a^i)^2$ is used because of $q_a^i = \{0,1\}$ for the transformation. By substituting Eq. (21) into the constraint function of Eq. (19), the following equation is derived.

$$H_{constraint} = -\sum_{i=1}^{N}\lambda_i\sum_{a=1}^{K}(q_a^i)^2 + 2\sum_{i=1}^{N}\lambda_i\sum_{\substack{a=1\\ b>a}}^{K} q_a^i q_b^i + \sum_{i=1}^{N}\lambda_i. \qquad (22)$$

As a result, the objective function of Eq. (18) and the constraint function of Eq. (22) are composed of the sum of the product of two variables $q_a^i q_a^j$, $q_a^i q_b^i$, or $(q_a^i)^2$.

Therefore, the QUBO coefficients can be obtained as follows.

$$\underset{\substack{\forall i=1\sim N, \forall j=i\sim N\\ \forall a=1\sim K, \forall b=a\sim K}}{Q((i,a),(j,b))} = \begin{cases} d(x_i,x_j) & (i<j, a=b) \\ -\lambda & (i=j, a=b) \\ 2\lambda & (i=j, a<b). \end{cases} \qquad (23)$$

Note that the third term in Eq. (22) can be ignored in the QUBO matrix because it does not interact with variables q.

In this paper, the formulation of Eq. (23) is called *Compact QUBO Definition (CQD)*. Algorithm 2 shows the pseudo code of the QUBO matrix generation that includes the one-hot constraint.

Algorithm 2 Generating a QUBO matrix that includes the one-hot constraint.

```
1: Initialize $k$ to the number of clusters
2: Initialize $n$ to the number of data points
3: Initialize $D$ to the normalized distance matrix of data
4: Initialize $\lambda$ to the result of subtracting $k$ from $n$
5: Let $Q$ be a new $nk \times nk$ QUBO matrix
6: for $i = 1$ to $n$ do
7:     for $j = i$ to $n$ do
8:         for $a = 1$ to $k$ do
9:             for $b = 1$ to $k$ do
10:                if $i < j$ and $a = b$ then
11:                    $Q((i,a),(j,b)) = D(i,j)$
12:                else if $i = j$ and $a = b$ then
13:                    $Q((i,a),(j,b)) = -\lambda$
14:                else if $i = j$ and $a < b$ then
15:                    $Q((i,a),(j,b)) = 2\lambda$
16:                end if
17:            end for
18:        end for
19:    end for
20: end for
```

Algorithm 3 Generating a QUBO matrix that does not include the one-hot constraint.

```
1: Initialize $k$ to the number of clusters
2: Initialize $n$ to the number of data points
3: Initialize $D$ to the normalized distance matrix of data
4: Let $Q$ be a new $nk \times nk$ QUBO matrix
5: for $i = 1$ to $n - 1$ do
6:     for $j = i + 1$ to $n$ do
7:         for $a = 1$ to $k$ do
8:             $Q((i,a),(j,a)) = D(i,j)$
9:         end for
10:    end for
11: end for
```

In CQD, QUBO coefficients are directly given in the QUBO matrix since their assignment can be decided in advance. Furthermore, the method of directly assigning coefficients can be executed faster than PyQUBO because the function call of `append`, which is required in PyQUBO, is not required in CQD. This technique can also be used in various languages to directly store QUBO coefficients in the QUBO matrix.

Moreover, SAEC that externally defines the constraint function does not need to calculate the coefficients of the constraint function as the QUBO coefficients. CQD for SAEC calculates only the coefficients of Eq. (18). Thus, the QUBO coefficients can be assigned as follows when SAEC is used.

$$\underset{\substack{\forall i=1\sim N-1, \forall j=i+1\sim N \\ \forall a=1\sim K,}}{Q((i,a),(j,a))} = d(x_i, x_j). \tag{24}$$

In the left hand side of Eq. (24), "a" of the first element and that of the second element are common. However, the notation cannot be changed from $((i,a),(j,a))$ to (i,j,a). This is because the coefficients of the QUBO matrix represent the interaction between qubits, and the size of the QUBO matrix is the square of the number of qubits. Algorithm 3 shows the pseudo code of the QUBO matrix generation that does not include the one-hot constraint. Conventionally, SA requires the large Lagrange multiplier for the constraint. However, SAEC does not require the Lagrange

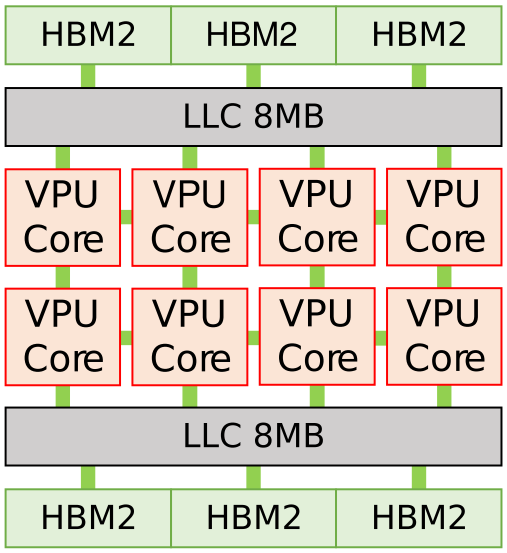


Figure 7: The block diagram of a vector engine.

multiplier because the constraint is satisfied by the operations with multiple bit-flips. As a result, the CQD used for SAEC in Algorithm 3 can be more computationally efficient than the CQD used for SA in Algorithm 2.

4 Evaluation

4.1 Experimental Environments

This section evaluates the proposed method in terms of the qualities of clustering results and the execution time.

To execute the proposed method of externally-defined constraint, *SX-Aurora TSUBASA (Aurora)* is used. Aurora consists of a CPU (Intel Xeon 6126) and a VE (NEC Vector Engine Type 10B) [21–24]. Figure 7 shows the block diagram of a VE [25]. A VE contains a vector processor (VPU), a 16 MB shared last-level cache, and six HBM2 memory modules. A VPU is a high-performance processor that can store many data in vector registers and execute operations simultaneously, where operations are called vector operations. In SA, the calculation of the energy after the bit-flip operation is the most computationally complex. Since this calculation is a matrix operation and can be accelerated by vector operations. The solver that executes SA and SAEC using Aurora is called *Aurora Simulated Annealing (ASA)*.

ASA also uses a parallel processing method [19] called *Parallel Tempering (PT)* [26]. PT allocates different temperatures to multiple processes. Each process executes SA or SAEC independently. Moreover, solutions are probabilistically exchanged between processes according to the temperature. The number of parallel temperature on a CPU is 12, and that on a VE is 8. The number of sweeps is set to 100. Inverse temperature $\frac{1}{T}$ is set to 0.1 when starting the solver, and increases until 50 at the end.

Table 1 shows clustering methods used in the experiments. This paper compares the proposed method with the conventional method, K-means, K-means++, and Spectral clustering. The conventional clustering of internally-defined constraint is called the conventional method. The conventional method minimizes the QUBO function that includes both the objective function and the constraint function. The annealing algorithms used in the experiments are ASA and D-Wave's *Leap Hybrid Sampler (Leap)*. ASA is executed on a CPU and a VE, respectively, as in the proposed method.

Table 1: Clustering methods used in the experiments.

Clustering method	Framework	Processor
The proposed method of the externally-defined constraint	Aurora simulated annealer	CPU
	Aurora simulated annealer	VE
The conventional method of the internally-defined constraint	Aurora simulated annealer	CPU
	Aurora simulated annealer	VE
	Leap hybrid sampler	QPU, CPU
K-means	Scikit-learn	CPU
K-means++	Scikit-learn	CPU
Spectral	Scikit-learn	CPU

Table 2: The data sets used for the evaluation.

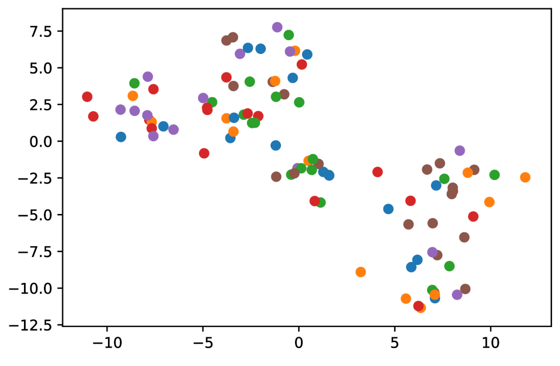
Data set	# of instances	# of classes
Iris	150	3
Wine	178	3
Breast cancer	683	2
Connectionist bench	208	2
Ionosphere	351	2
Seeds	210	3

QA cannot be executed if the problem size is large because the number of qubits is limited. A Leap hybrid sampler combines QA on a QPU (Quantum Processing Unit) and conventional computation on a CPU for larger problems instead of using only QA. Leap are used through the dwave-ocean-sdk v2.1.1 Python interface. The other methods to be compared with the proposed method are quasi-optimal clustering methods such as K-means, K-means++, and Spectral clustering that are executed on a CPU by using scikit-learn v0.23.2 [27]. K-means and K-means++ are performed to minimize the approximate objective function shown in Eq. (3). Spectral clustering is a method that can be used for clustering complicated data, although it does not minimize the distance between data as in the functions of Eqs. (1) and (3).

QUBO matrices of combinatorial clustering for annealing algorithms are generated by CQD. PyQUBO v0.4.0 is compared to the QUBO generation using CQD. QUBO matrices are generated on a CPU.

The data sets used to evaluate all these five methods are an artificial data set generated by `make_blobs` in scikit-learn [27], toy data sets by loading from the `sklearn.datasets` module [27], and real data sets from the UCI Machine Learning Repository [28]. All the artificial data sets are generated with the standard deviation of the clusters of 1.5. Table 2 shows the data sets used in the experiments. The toy data sets are closer to real data sets than artificial ones. In Table 2, Iris and Wine are the toy data sets, and the others are the real data sets.

The quality of a clustering result is evaluated by Cost as the value of an exact objective function shown in Eq. (1). The distances used to calculate Cost are normalized by the maximum distance, and the values of Cost are normalized by the mean Cost values of the proposed method. The execution time is measured by omitting the file reading and writing times and the communication time to the D-Wave Cloud server. All experiments are run in Python except for the process of solving the QUBO problem. Since the execution time tends to vary in the Python environment, all the experiments are carried out 100 times, and the mean value and the error bar with the standard deviation are presented.


(a) The conventional method.

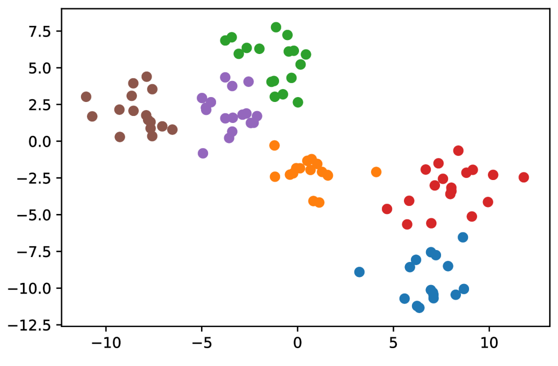

(b) The proposed method.

Figure 8: The results of combinatorial clustering.

4.2 Evaluation of Externally-Defined One-Hot Constraint Clustering

4.2.1 Evaluation of Quality

Figure 8 shows the results of the conventional method and the proposed method, both on a CPU. The number of data points N is 100, and the number of clusters K is 6. In Figure 8 (a), all the data points belong to a single cluster. This is because SA can minimize the constraint function in the QUBO function. When the QUBO function contains both the objective function and the constraint function, the coefficients of the constraint function become very large. Since the influence of the objective function becomes relatively small, the conventional method succeeds in minimizing the constraint function, but fails in minimizing the objective function. As a result, the conventional method fails in clustering the data points as shown in Figure 8 (a). Figure 8 (b) shows that the proposed method succeeds in clustering the data points. This is because SAEC minimizes the objective function independently from the constraint.

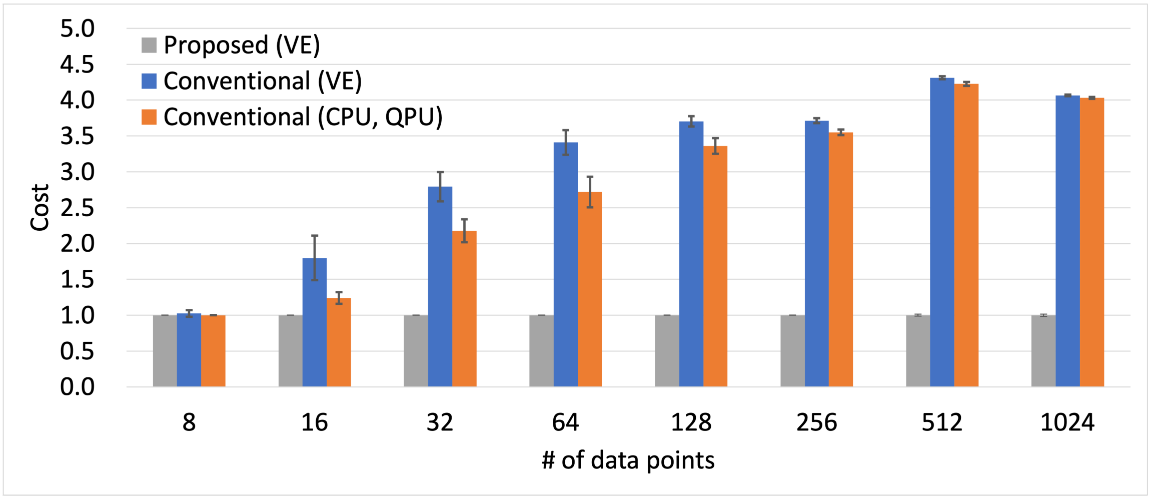
Figure 9 compares the proposed method with the conventional method in aspect of Cost. In Figure 9, *Conventional* indicates the conventional method that includes the constraint function in a QUBO matrix. *Proposed* indicates the proposed method that externally defines the constraint. The proposed method is executed by ASA on a VE. The conventional method is executed by ASA on a VE and Leap Hybrid of both a CPU and a QPU. Figures 9 (a) and (b) are the results using the artificial data sets. Figure 9 (a) changes the number of data points when the number of clusters is 3. Figure 9 (b) changes the number of clusters when the number of data points is 256.

From these figures, the proposed method gives the best quality results regardless of the number of data points and clusters. This is because the conventional method using the Lagrange multiplier makes it difficult to successfully perform combinatorial clustering due to the limitation of precision on the hardware. Figure 9 (a) also shows that the difference between the conventional and the proposed methods increases as the problem size increases. Cost of the proposed method is less than a quarter of Cost of the conventional method when the number of data points is 256 or more. This is because the conventional method requires the large Lagrange multiplier for the large number of data points.

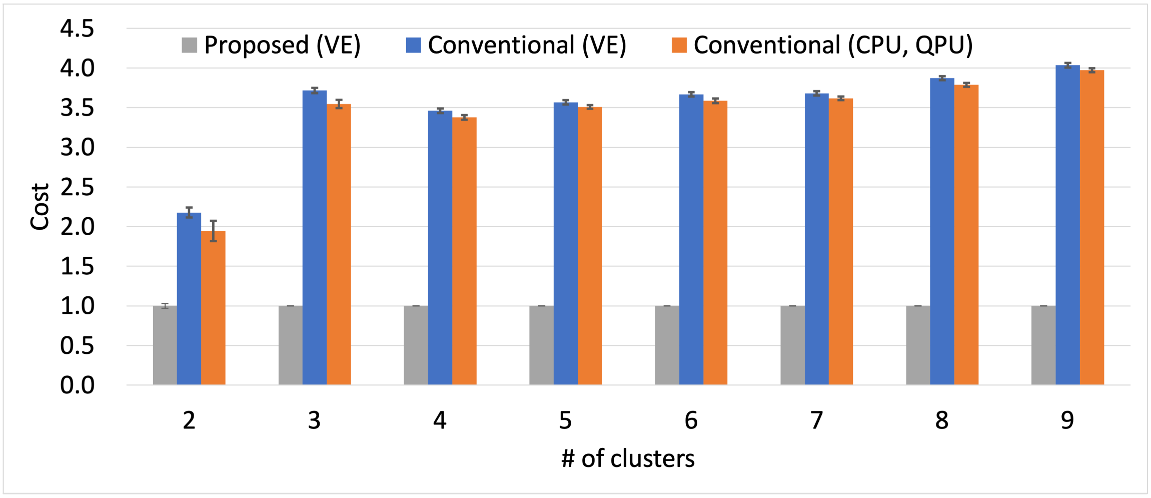
Figure 9 (c) shows the results using the toy data sets and the real data sets. This figure shows that Cost of the proposed method is lower than that of the conventional method in all cases. Cost of the proposed method is lower than half of Cost of the conventional method on average. This is because the proposed method optimizes the objective function separately from the constraint. Thus, the proposed method can perform high quality clustering not only on the artificial data but also on the real data.

Moreover, the error bars of the proposed method are shorter than those of the conventional methods. This is because the proposed method achieves the lowest Cost in almost all trials. Since the proposed method has a little variation of Cost, high quality clustering results are obtained stably.

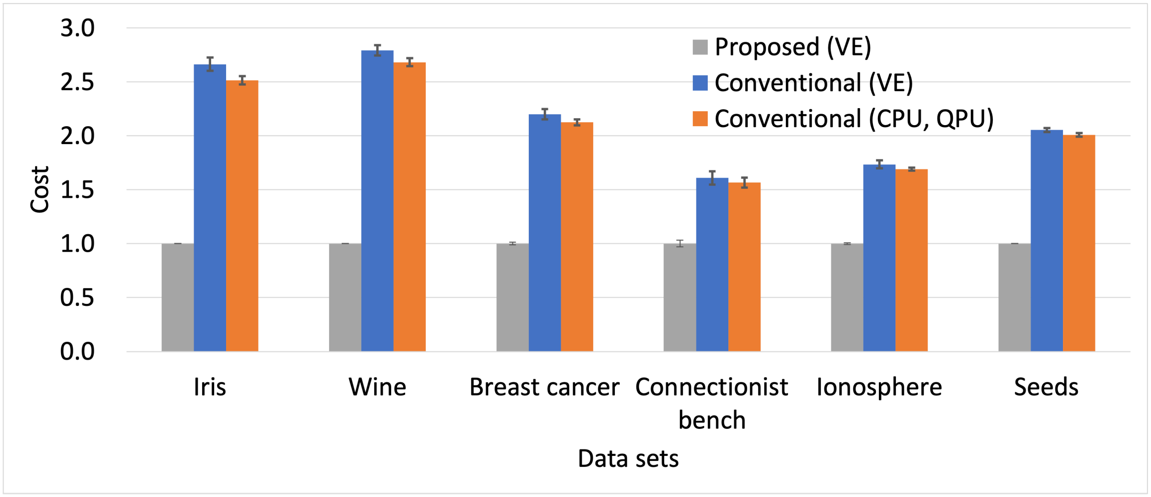
Figure 10 compares the proposed method with the quasi-optimal clustering methods in aspect



(a) The number of data points.



(b) The number of clusters.



(c) Toy data sets and real data sets.

Figure 9: Cost compared to conventional methods.

of Cost. All the quasi-optimal clustering methods are executed on a CPU. Figures 10 (a) and (b) are the results using the artificaial data sets. Figure 10 (a) changes the number of data points when the number of clusters is 3. Figure 10 (b) changes the number of clusters when the number of data points is 256.

These figures show that Cost of the proposed method is lower than or equal to that of K-means, K-means++, and Spectral clustering. The proposed method can find the optimal solution if the objective function can be minimized. However, the optimal solution of the K-means approximate objective function shown in Eq. (3) is not necessarily equivalent to that of the exact objective function shown in Eq. (1). As a result, the proposed method achieves the highest quality clustering.

Figure 10 (c) is the result using the toy data sets and the real data sets. This figure also shows that Cost of the proposed method is lower than or equal to that of K-means, K-means++, and Spectral clustering. Thus, even for real-world data, the proposed method has the potential to achieve the best clustering results by minimizing the value of the exact objective function of clustering. These figures also indicate that Cost of Spectral clustering is much higher than the proposed method, K-means, and K-means++. The smaller the value of the objective function in Eq. (1), the smaller Cost becomes. Spectral clustering is a clustering method that utilizes graphs' connected structure [29] and has a different objective function from Eq. (1). Therefore, Spectral clustering does not minimize the distance between data points within a cluster. On the other hand, there are data with complicated distributions that cannot perform appropriately clustering by methods that reduce the distance between data, such as combinatorial clustering. Spectral clustering can perform high-quality clustering for such complicated data.

Moreover, the variance is very small in all methods. This is because the proposed method stably obtains the exact optimal solution, while the quasi-optimal clustering methods stably obtain the sub-optimal solution.

4.2.2 Evaluation of Execution Time

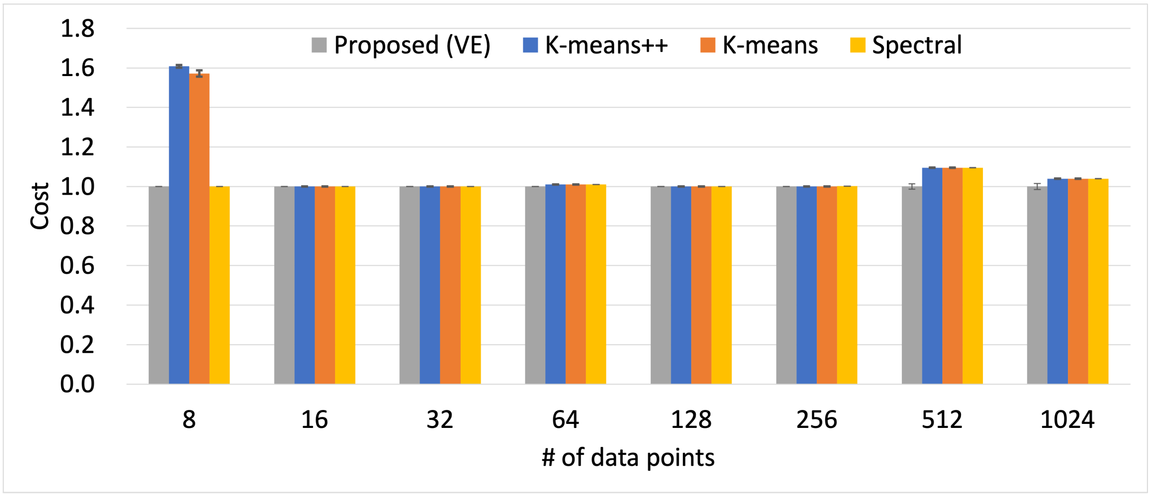
Figure 11 shows the execution time of the clustering methods with the different number of data points. Note that the execution time includes the QUBO generation time. The number of data points is varied from $N = 8$ to $1,024$ and the number of clusters K is 3. Figure 11 shows that the execution times of the conventional method using Leap is almost constant when the number of data is 256 or less. This is because Leap may use a QPU when the number of data is small. Since Leap starts using CPU when the number of data points becomes large, the execution time may rapidly increase.

Besides, the execution times of K-means, K-means++, and Spectral clustering are shorter than that of annealing-based clustering on all the data sets. This is because clustering methods using annealing algorithms requires N^2 computation for distance calculation. Quasi-optimal clustering reduces the computational complexity by calculating the distance using the centroid within a cluster. K-means is the fastest because the distance calculation is reduced to about NK.

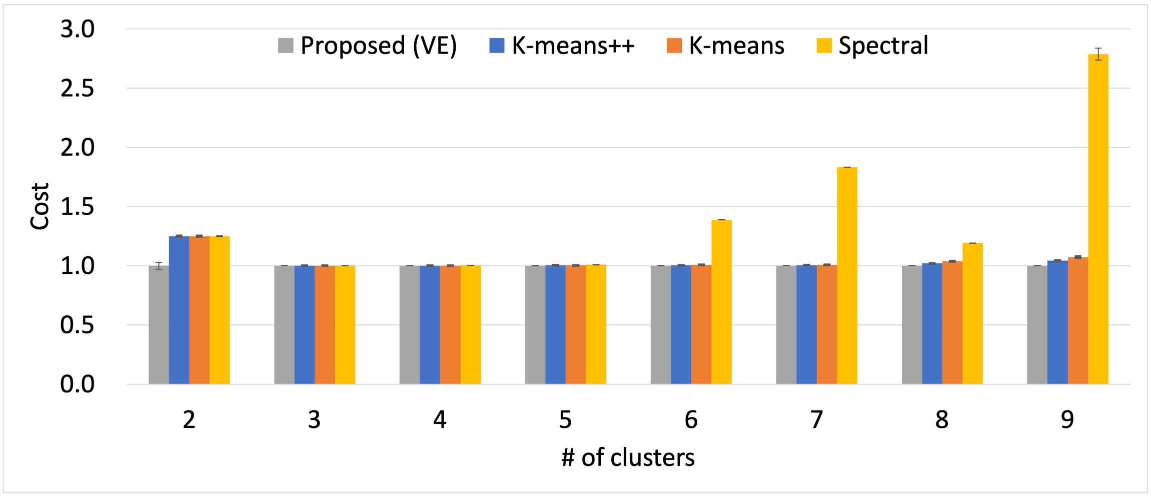
Moreover, the execution time of a VE becomes shorter than that of a CPU when the number of data is large. When the number of data points is 1,024, the execution time of the proposed method on a VE is 18.2 times shorter than that on a CPU. This is because a VE uses vector operations to calculate the energy of the QUBO function. In the case of small data, the vector length is short, and the calculation on a VE becomes inefficient. In the case of large data, the vector register is filled. Therefore, a VE efficiently works with a long vector length.

Furthermore, the execution time of the proposed method is shorter than that of the conventional method in the cases of both a CPU and a VE when the number of data is large. When the number of data is 1,024, the execution time of the proposed method on a VE is 1.4 times shorter than that of the conventional method on a VE. This is because the decrease in the calculation of the coefficients of the constraint function is larger than the increase in the operations with multiple bit-flips time in SAEC.

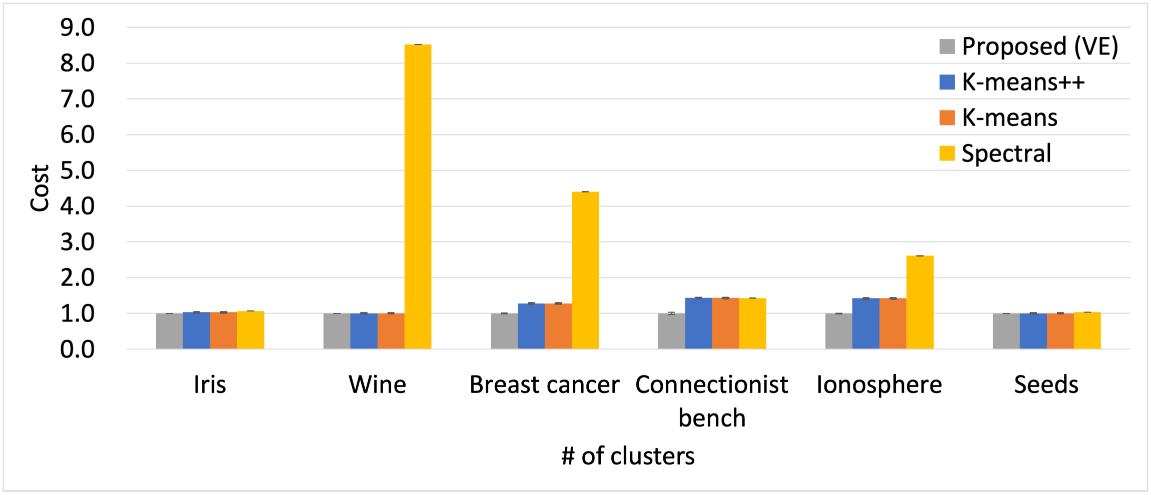
To confirm the reason why the execution time of the proposed method is shorter than that of the conventional method when the number of data is large, Figure 12 shows the breakdown of the execution time on a VE. The x-axis indicates the number of data points. For each data points in



(a) The number of data points.



(b) The number of clusters.



(c) Toy data sets and real data sets.

Figure 10: Cost compared to quasi-optimal methods.

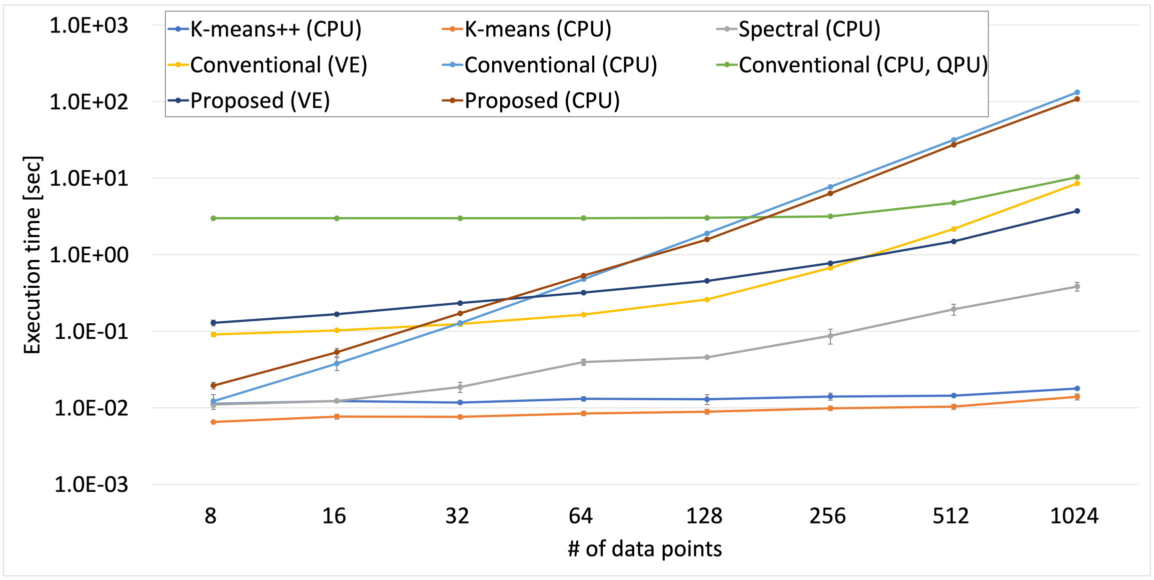


Figure 11: Execution time of clustering methods with the different number of data points.

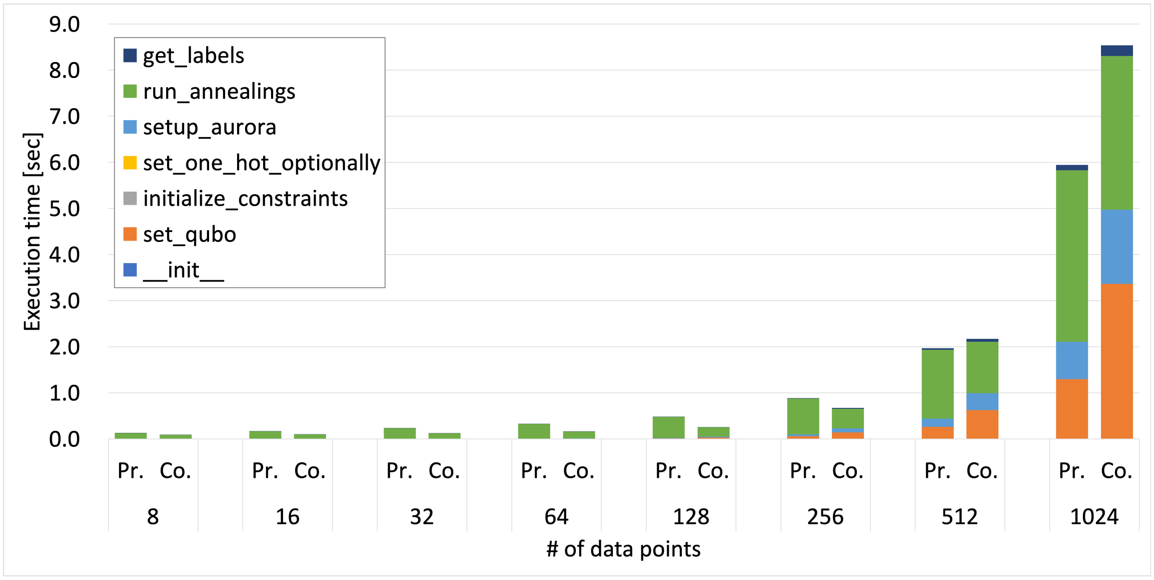


Figure 12: Breakdown of combinatorial clustering on a VE with the different number of data points.

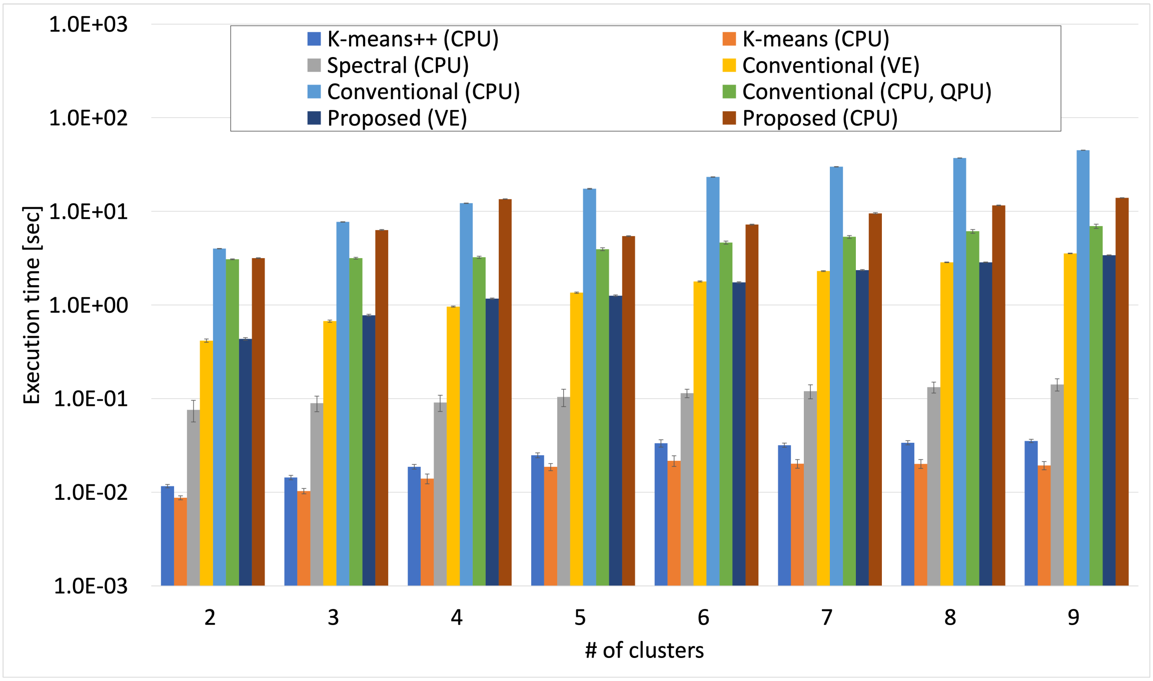


Figure 13: Execution time of clustering methods with the different number of clusters.

the x-axis, *Pr.* is the proposed method, and *Co.* is the conventional method. The y-axis indicates the execution time and shows the breakdown of the execution into the individual processes for each data. Here, `run_annealings` indicates the execution time of annealing, and `set_qubo` is the execution time to generate the QUBO matrix. When the number of data is 1,024, the annealing time of the proposed method is 1.1 times longer than that of the conventional method. This is because the proposed method needs to satisfy the one-hot constraint by the bit-flip operations. The proposed method executes the process of satisfying the constraint at every sweep. However, the QUBO matrix generation time of the proposed method is 2.6 times shorter than that of the conventional method. This is because the proposed method does not include the constraint in the QUBO matrix, resulting in the reduction in the generation process of the QUBO matrix.

Figure 13 shows the execution time of the clustering methods with the different number of clusters. The number of clusters is varied from $K = 2$ to 9, and the number of data points N is 256. Figure 13 shows that the execution times of K-means, K-means++, and Spectral clustering are shorter than that of annealing-based clustering on all the data sets. This is because quasi-optimal clustering reduces the computational complexity by calculating the distance using the centroid within a cluster. The execution times using a VE are much shorter than those using a CPU in the both cases of the conventional and proposed methods. In this experiment, the execution time of the proposed method on a VE is 5.3 times shorter than that on a CPU on average. The execution time of the conventional method on a VE is also 12.3 times shorter than that on a CPU on average. This is because the vector length is long enough when the number of data is 256.

To further investigate the difference in the execution times between the proposed and conventional methods, Figure 14 shows the breakdown of the execution time on a VE. The x-axis indicates the number of clusters. In Figure 14, the annealing time of the proposed method is longer than that of the conventional method. However, the QUBO matrix generation time of the proposed method is shorter than that of the conventional method. This is because in the proposed method, the calculation to satisfy the constraint is not included in the QUBO matrix generation time, but included in the annealing time. The number of calculations to generate the QUBO matrix is N^2K in the proposed method, while N^2K^2 in the conventional method. Therefore, when the number of clusters is large,

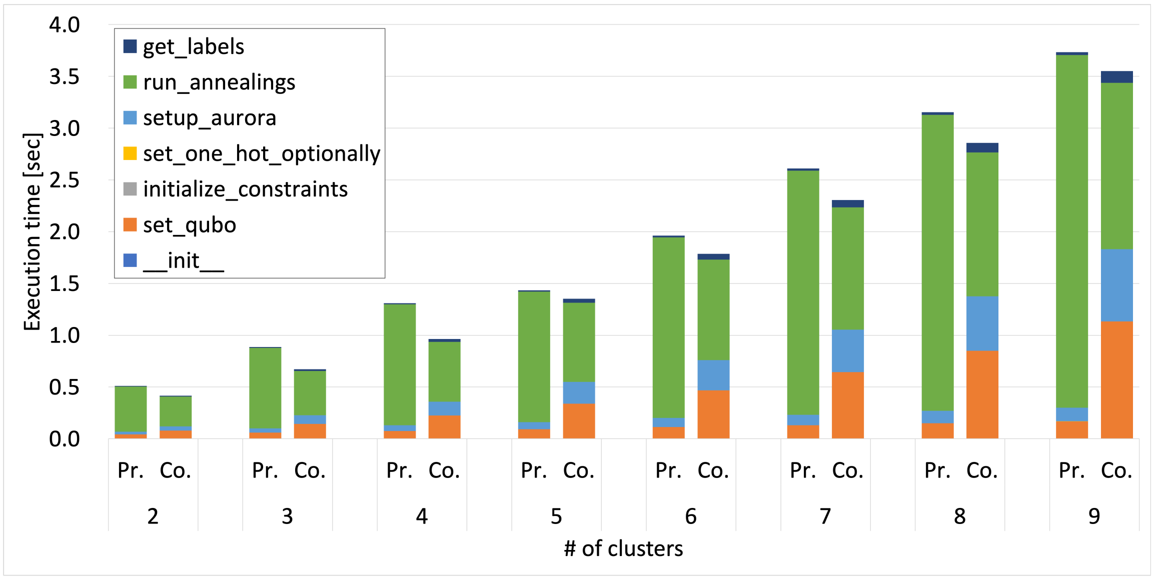


Figure 14: Execution time of combinatorial clustering on a VE with the different number of clusters.

the execution of the proposed method is expected to be faster than that of the conventional method.

4.3 Evaluation of Externally-Defined One-Hot Constraint Clustering by using the Multiple Traveling Salesman Problem

This subsection evaluates the proposed method by using one of practical applications, *the Multiple Traveling Salesman Problem (MTSP)* [30]. MTSP might be useful for applications such as calculation of tsunami evacuation routes [25,31,32]. Given a distance matrix of N cities and m salesmen, MTSP needs to appropriately classify cities that each salesman visits. To minimize the total tour length for all salesmen, clustering is used to assign cities to m salesmen, which has been actively studied recently [33, 34]. The procedure of MTSP is as follows, which is also shown in Figure 15.

1. Cities are grouped by clustering so that they are close together.

2. Calculate m centroids $\mu_1, \ldots, \mu_m$ for each cluster.

3. Calculate the centroid of $\mu_1, \ldots, \mu_m$, called μ^*.

4. Add the point closest to μ^* to each cluster as the starting point for all salesmen.

5. Solve *the Traveling Salesman Problem (TSP)* within each new cluster.

The proposed annealing-based clustering method is compared with the conventional annealing-based clustering method and K-means++. The Google OR-Tools heuristic solver is used to solve TSP [35]. Thirteen data sets with the different numbers of cities from TSPLIB are used as experimental data [36]. The number of salesmen is set to four in all experiments as the number of clusters. The evaluation metrics are Cost of clustering and the tour length of MTSP. Cost and the tour length are normalized by the mean values of each in the proposed method.

Figure 16 shows Cost of clustering in MTSP. Cost of the proposed method is lower than that of the conventional method in all cases. This is because the proposed method optimizes the objective function independently from the constraint. As a result, the proposed method can appropriately group cities that are close to each other. Cost of the proposed method show a higher quality than K-means++ in all cases. This is because the proposed method optimizes the exact objective function

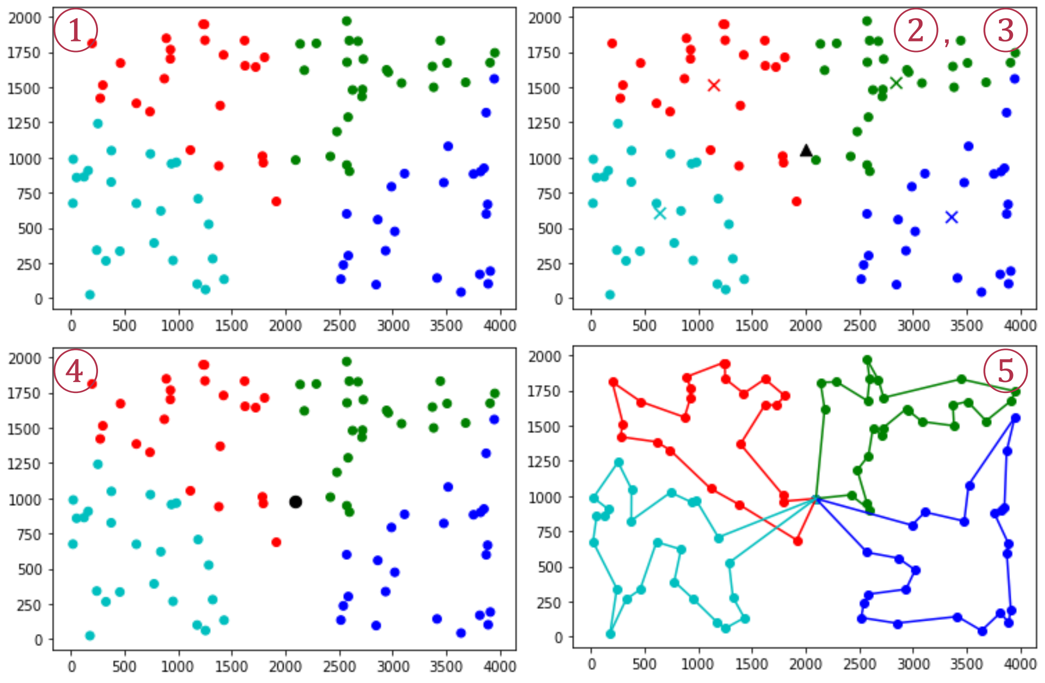


Figure 15: The procedure of MTSP.

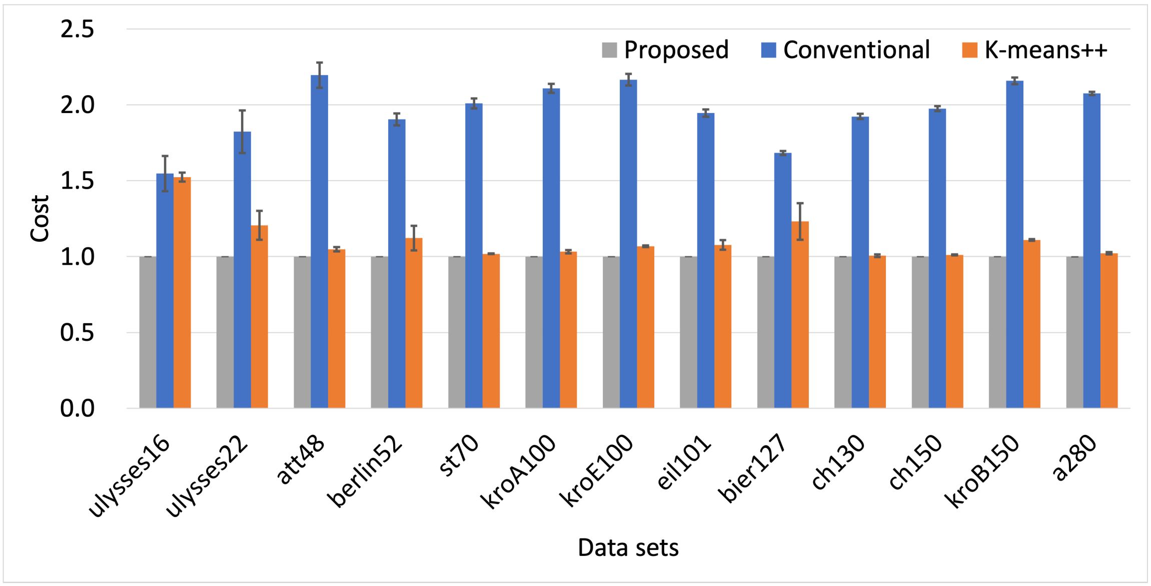


Figure 16: Cost of clustering methods in MTSP.

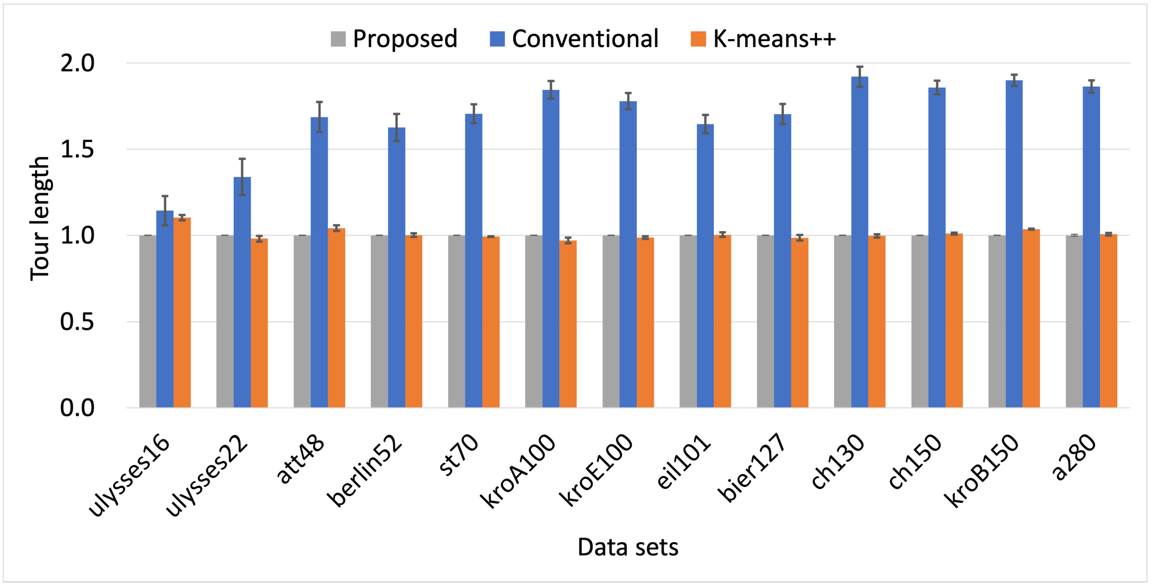


Figure 17: The tour length of MTSP.

of Eq. (1). Therefore, the proposed clustering method is useful even for practical data used in real problems.

Figure 17 shows the tour length of MTSP. Basically, the smaller Cost is, the shorter the tour length tends to be. The tour length using the proposed method is shorter than that using the conventional method in all cases. This is because the proposed method finds the tour length between cities that are close to each other. Moreover, the tour length of the proposed method is shorter than the tour length of K-means++ in 7 out of 13 cases.

To further investigate the significant difference in the tour length between the proposed method and K-means++, the F-test and Welch's t-test are conducted as a statistical significance test [37]. The F-test is a test for the homogeneity of the variances of two experimental results. The t-test is a test to examine the significant difference between the results of two experiments, and Welch's t-test is used especially when the variances are unequal. Before conducting the t-test, the F-test is conducted to examine the equality of their variances. As a result, the p-values of the F-test are lower than the 0.05 level of significance in all cases. Since all the variances of the tour length in the proposed method are not equal to those in K-means++, Welch's t-test is performed to verify the significance of the differences, as shown in Table 3.

Table 3 shows the mean, standard deviation of the tour length indicated by *SD*, and results of Welch's t-test. *Prop.* indicates the proposed method and *K++* indicates K-means++. The smaller values of the mean and standard deviation between the proposed method and K-means++ are shaded. The p-values of the t-test less than 0.05 are also shaded.

From Table 3, the standard deviations of the proposed method are smaller than those of K-measn++ in all the cases. The proposed method provides the results of the tour length with a low variance. This is because the proposed clustering method obtains the same clustering results in almost all trials. When identical clustering results are obtained, identical travel routes are easily obtained because the set of cities to solve TSP is the same. Since the variance of Cost in K-means++ is larger than that in the proposed method, the variance of the tour length in K-means++ is also larger than that in the proposed method.

The p-value of Welch's t-test is lower than the 0.05 level of significance in all the data sets except for berlin52. Here, there is no significant difference between the proposed method and K-means++ on berlin52. On the other hand, there is a significant difference between the proposed method and K-means++ on the other data sets. In 6 out of these 12 data sets, the mean tour length of the

Table 3: The mean, standard deviation of the tour length, and results of Welch's t-test.

		Data sets						
		ulysses16	att48	eil101	ch150	kroB150	a280	berlin52
Mean	Prop.	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	K++	1.103	1.042	1.004	1.010	1.036	1.006	1.001
SD	Prop.	1.554E-15	1.110E-15	0.000E+00	4.441E-16	8.882E-16	5.557E-03	1.221E-15
	K++	1.605E-02	1.591E-02	1.371E-02	5.459E-03	3.749E-03	7.230E-03	1.076E-02
P-value		2.993E-82	1.974E-46	3.719E-03	7.941E-34	1.354E-99	4.291E-10	2.343E-01

		Data sets					
		ulysses22	st70	kroA100	kroE100	bier127	ch130
Mean	Prop.	1.000	1.000	1.000	1.000	1.000	1.000
	K++	0.981	0.993	0.971	0.987	0.987	0.998
SD	Prop.	5.005E-16	1.332E-15	1.110E-15	6.661E-16	1.554E-15	1.332E-15
	K++	1.632E-02	1.892E-03	1.632E-02	7.400E-03	1.599E-02	9.878E-03
P-value		4.768E-20	4.425E-57	1.759E-32	2.232E-32	4.897E-13	1.556E-02

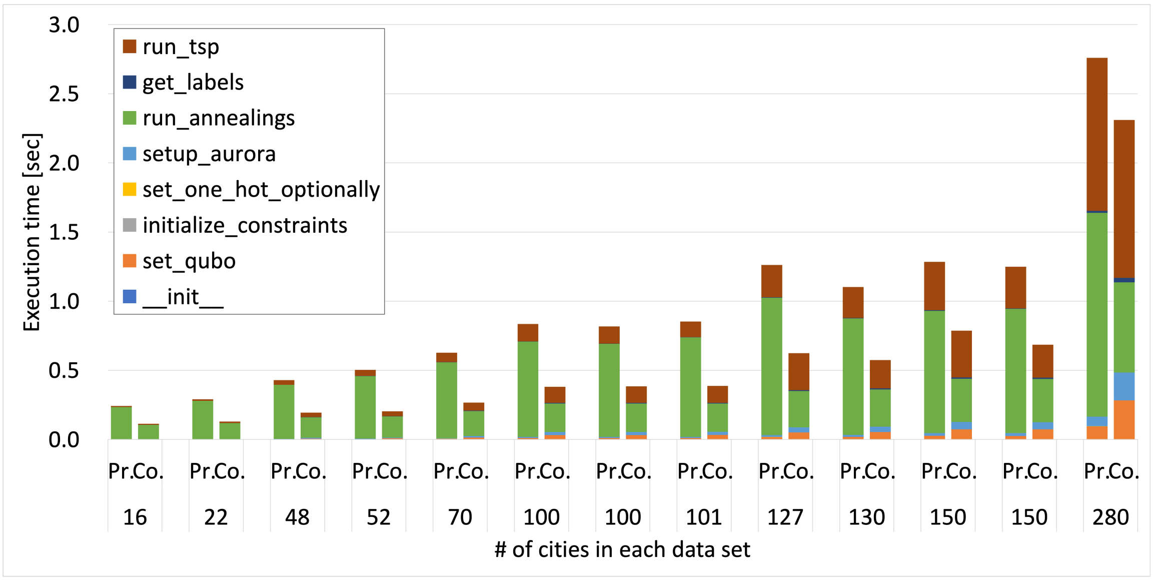


Figure 18: Execution time of MTSP.

proposed method is shorter than that of K-means++. There are two reasons that the tour length of all data sets using the proposed method is not always shortest. One is that the heuristic solver used to solve TSP in a cluster cannot find the optimal solution. This is because TSP is solved by a heuristic method, which does not necessarily provide an optimal solution. The second reason is that the clustering cities close to each other may not shorten the tour length. The tour length of the proposed method is almost as good as those of K-means++, even though the tour length is not the shortest.

To clarify the fraction of the proposed clustering to the total execution of MTSP, Figure 18 shows execution time of MTSP using annealing-based clustering methods. The x-axis indicates the number of cities in each data set. The y-axis indicates the execution time and shows the breakdown of the execution into the individual processes for each data. Here, `run_tsp` indicates the execution time to solve TSP.

When the number of cities is 16, the annealing time occupies the longest time in the whole process, which is 91.9% in the conventional method and 96.6% in the proposed method. However, when the number of cities is 280, the annealing times of the conventional method and the proposed method occupy 28.4% and 53.4% of the total execution time, respectively. This is because the

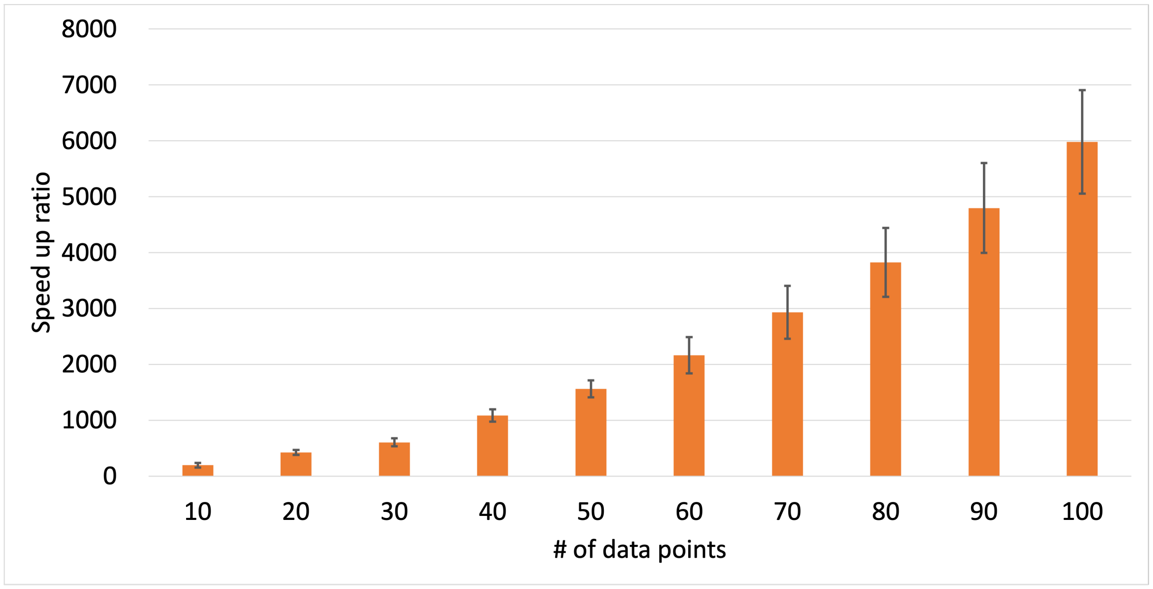


Figure 19: Speed up ratio of the QUBO generation.

execution time of simulated annealing is determined by the number of sweeps and does not change significantly even when the problem size increases. On the other hand, the time to solve TSP using the conventional method rapidly increases from 6.8% to 49.0% when the number of cities increases from 16 to 280. Similarly, the time to solve TSP using the proposed method also increases from 3.2% to 40.1%. This is because TSP is a combinatorial optimization problem as well as combinatorial clustering, and it takes time to find the optimal solution from many possible solutions. To achieve the practical use of MTSP using the proposed method, the execution time to solve TSP needs to be accelerated.

4.4 Evaluation of the Compact QUBO Definition

Finally, the speed up of the QUBO generation time by using CQD is evaluated. Figure 19 shows the speed up ratios of the QUBO generation. The number of data is varied from $N = 10$ to 100, and the number of clusters K is 2. The x-axis indicates the number of data points, and the y-axis indicates the speed up ratios of the proposed CQD implementation to the conventional PyQUBO implementation in the conventional method. Note that the QUBO generation time does not include the time to generate the distance matrix.

The speed up ratio becomes huge as the problem size increases. Moreover, the QUBO matrix generation time of the proposed CQD implementation is almost 5,000 times shorter than that of the conventional PyQUBO implementation when the number of data points is 90 or more. This is because the number of QUBO coefficients is correlated with the square of the number of data points. The implementation using PyQUBO takes a very long time to generate the QUBO matrix. In the proposed CQD implementation, the execution time is reduced by giving the QUBO coefficients directly to the QUBO matrix. The variance of the speed up ratio increases as the number of data points increases. This is because the variance of QUBO generation time in both PyQUBO and the proposed method increases as the number of data increases.

5 Related Work

In recent years, various clustering methods based on annealing algorithms have been developed. They are formulated as QUBO problems to use annealing algorithms. Since the QUBO problem is based on the Ising model, clustering methods expressed in the QUBO problem are often referred to as *Ising clustering*.

Bauckhage et al. proposed *Adiabatic quantum kernel k=2 means clustering* that is a binary clustering using the kernel method [38]. Although this method is limited to 2 clusters, it has a wide application range because the kernel trick can be applied to various data structures. Adiabatic quantum kernel k=2 means clustering does not require a constraint function in the QUBO function. This is because the method is restricted to the case when the number of clusters is 2, like Eq. (16). Kernel k=2 means clustering can be extended to the case when the number of clusters is 3 or more by using one-hot encoding. When the one-hot encoding is used for representing qubits, a one-hot constraint is required for a data point to be assigned to one cluster. Therefore, the proposed method will be useful to satisfy the one-hot constraint.

Ising clustering has also been applied to semi-supervised clustering. Cohen et al. proposed Ising-based *Constrained clustering* that is Ising clustering with various types of supervisory information [39]. In Ising-based constrained clustering, the supervisory information is given by the coefficients of the QUBO matrix. Therefore, Ising-based constrained clustering can easily provide several different supervisory information simultaneously. Cohen et al. also proposed Ising-based *consensus clustering* that is a method for obtaining a new optimal clustering result by integrating multiple known clustering results [40]. Ising-based consensus clustering also uses the one-hot constraint. Besides, Arthur et al. proposed Ising-based *Balanced K-means clustering* that is also a kind of semi-supervised clustering using the Ising model [41, 42]. Balanced K-means clustering performs clustering in which all clusters have the same number of elements. It is claimed to be less computational complexity than similar conventional methods. Ising-based balanced K-means clustering also uses the one-hot constraint. Their Ising-based semi-supervised clustering methods combine the objective function and the one-hot constraint function into the QUBO function. The Lagrange multiplier must be set to a large enough value to satisfy the one-hot constraint. However, conventional annealers cannot set the large Lagrange multiplier due to the limitation of the bit precision. In practice, Cohen et al. and Arthur et al. set a small Lagrange multiplier. The obtained solutions include many solutions that violate the constraint, and the solutions that satisfy the constraint is adopted. However, there is a problem that this method requires a large number of solutions to obtain solutions that satisfy the constraint. It takes a long time to obtain the optimal solution. On the other hand, our proposed method is very efficient because it can obtain only solutions that satisfy the constraint with a high probability.

Moreover, Ising-based semi-supervised clustering methods also have constraints other than the one-hot constraint. This is because kinds of supervisory information are represented as the constraints. The QUBO function requires the multiple Lagrange multipliers to combine the multiple constraint functions. Optimization for clustering becomes very difficult due to the complexity of the search space. Our proposed method can be applied to other constraints by modifying the operations with multiple bit-flips. Since the proposed method satisfies multiple constraints simultaneously, the search space is limited to the objective function. As a result, the proposed method will improve the quality of clustering.

6 Conclusions

In recent years, combinatorial clustering has drawn much attention to obtain high quality clustering results. However, the method of solving combinatorial clustering using annealing is difficult to optimize the objective function because the objective function and the constraint function are combined by the Lagrange multiplier method.

This paper has proposed combinatorial clustering to overcome the limitation of the method of the Lagrange multiplier. The proposed method uses the QUBO solver with the externally-defined constraint function. Moreover, this paper has also proposed a technique to shorten the execution time of the QUBO generation by deriving the regularity of QUBO coefficients.

The evaluation results obtained by using the artificial data sets and real data sets show that the proposed method can improve the quality of clustering results compared with the conventional method that includes the constraint function in a QUBO matrix. The execution time of the proposed method is also shorter than that of the conventional method when the problem size is large. This is because the proposed method reduces the generation process of the QUBO matrix. Furthermore, the proposed method is evaluated using the multiple traveling salesman problem. The tour length using the proposed method is shorter than that using the conventional method in all cases and than that using K-means++ in 6 out of 12 cases with a significant difference. Moreover, the proposed CQD implementation shows almost 6,000 times faster execution of the QUBO matrix generation than the conventional technique.

The proposed method has been designed for digital annealers because the proposed method requires the operations with multiple bit-flips on digital computers. Ohzeki proposed an alternative approach to optimize the QUBO function that includes a constraint function on a quantum annealer [43]. Ohzeki used the Hubbard-Stratonovich transformation for the one-hot constraint function to remove the interaction between different qubits. As future work, it is necessary to study the method to improve the quality of combinatorial clustering on a quantum annealer. Furthermore, the detailed evaluation and analysis of the proposed method are necessary by applying the externally-define approach to other Ising clustering methods and data sets.

Acknowledgments

This research was partially supported by MEXT Next Generation High-Performance Computing Infrastructures and Applications R&D Program, entitled ”R&D of A Quantum-Annealing-Assisted Next Generation HPC Infrastructure and its Applications,” Grants-in-Aid for Scientific Research (A) #19H01095, and Grants-in-Aid for Scientific Research (C) #20K11838.

References

[1] Masahito Kumagai, Kazuhiko Komatsu, Fumiyo Takano, Takuya Araki, Masayuki Sato, and Hiroaki Kobayashi. Combinatorial Clustering Based on an Externally-Defined One-Hot Constraint. In *2020 Eighth International Symposium on Computing and Networking (CANDAR)*, pages 59–68. IEEE, 2020.

[2] M.S. Levin. Combinatorial clustering: Literature review, methods, examples. *Jounal of Communications Technology and Electronics*, 60:1403–1428, 2015.

[3] Anil K Jain. Data clustering: 50 years beyond K-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[4] Vaibhaw Kumar, Gideon Bass, Casey Tomlin, and Joseph Dulny. Quantum annealing for combinatorial clustering. *Quantum Information Processing*, 17(2):39, 2018.

[5] John E Dorband. Extending the d-wave with support for higher precision coefficients. *arXiv preprint arXiv:1807.05244*, 2018.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[7] Kenichi Kurihara, Shu Tanaka, and Seiji Miyashita. Quantum annealing for clustering. *arXiv preprint arXiv:1408.2035*, 2014.

[8] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[9] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[10] Endre Boros, Peter L Hammer, and Gabriel Tavares. Local search heuristics for quadratic unconstrained binary optimization (QUBO). *Journal of Heuristics*, 13(2):99–132, 2007.

[11] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[12] Steven S Skiena. *The algorithm design manual.* Springer International Publishing, 2020.

[13] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[14] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse Ising model. *Physical Review E*, 58(5):5355, 1998.

[15] Giuseppe E Santoro and Erio Tosatti. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *Journal of Physics A: Mathematical and General*, 39(36):R393, 2006.

[16] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, Richard Harris, Andrew J Berkley, Jan Johansson, Paul Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.

[17] Giuseppe E Santoro, Roman Martoňák, Erio Tosatti, and Roberto Car. Theory of quantum annealing of an Ising spin glass. *Science*, 295(5564):2427–2430, 2002.

[18] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach.* Elsevier, 2011.

[19] Fumiyo Takano, Motoi Suziki, Yuki Kobayashi, and Takuya Araki. QUBO solver for combinatorial optimization problems with constraints. Technical Report 4, NEC Corporation, nov 2019.

[20] Kotaro Tanahashi, Shinichi Takayanagi, Tomomitsu Motohashi, and Shu Tanaka. Application of Ising Machines and a Software Development for Ising Machines. *Journal of the Physical Society of Japan*, 88(6):061010, 2019.

[21] Kazuhiko Komatsu, Shintaro Momose, Yoko Isobe, Osamu Watanabe, Akihiro Musa, Mitsuo Yokokawa, Toshikazu Aoyama, Masayuki Sato, and Hiroaki Kobayashi. Performance evaluation of a vector supercomputer SX-aurora TSUBASA. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 685–696. IEEE, 2018.

[22] Mitsuo Yokokawa, Ayano Nakai, Kazuhiko Komatsu, Yuta Watanabe, Yasuhisa Masaoka, Yoko Isobe, and Hiroaki Kobayashi. I/O Performance of the SX-Aurora TSUBASA. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 27–35, 2020.

[23] Ilya V Afanasyev, Vadim V Voevodin, Vladimir V Voevodin, Kazuhiko Komatsu, and Hiroaki Kobayashi. Analysis of Relationship Between SIMD-Processing Features Used in NVIDIA GPUs and NEC SX-Aurora TSUBASA Vector Processors. In *International Conference on Parallel Computing Technologies*, pages 125–139. Springer, 2019.

[24] Ilya V Afanasyev, Vladimir V Voevodin, Kazuhiko Komatsu, and Hiroaki Kobayashi. VGL: a high-performance graph processing framework for the NEC SX-Aurora TSUBASA vector architecture. *Journal of Supercomputing*, 2021.

[25] Akihiro Musa, Takashi Abe, Takumi Kishitani, Takuya Inoue, Masayuki Sato, Kazuhiko Komatsu, Yoichi Murashima, Shunichi Koshimura, and Hiroaki Kobayashi. Performance Evaluation of Tsunami Inundation Simulation on SX-Aurora TSUBASA. In *International Conference on Computational Science*, pages 363–376. Springer, 2019.

[26] Robert H. Swendsen and Jian-Sheng Wang. Replica Monte Carlo Simulation of Spin-Glasses. *Phys. Rev. Lett.*, 57:2607–2609, Nov 1986.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Dheeru Dua and Casey Graff. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml, 2017.

[29] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[30] Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *omega*, 34(3):209–219, 2006.

[31] K. Komatsu, T. Kishitani, M. Sato, A. Musa, and H. Kobayashi. Search Space Reduction for Parameter Tuning of a Tsunami Simulation on the Intel Knights Landing Processor. In *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 117–124, 2018.

[32] T. Kishitani, K. Komatsu, M. Sato, A. Musa, and H. Kobayashi. Importance of Selecting Data Layouts in the Tsunami Simulation Code. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 830–837, 2020.

[33] Zhanqing Lu, Kai Zhang, Juanjuan He, and Yunyun Niu. Applying K-means Clustering and Genetic Algorithm for Solving MTSP. pages 278–284, 10 2016.

[34] Hirokazu Watanabe, Tsutomu Ono, Akihiro Matsunaga, Akihiro Kanagawa, and Hiromitsu Takahashi. Multiple Traveling Salesman Problems Using the Fuzzy c-means Clustering. *Journal of Japan Society for Fuzzy Theory and Systems*, 13(1):119–126, 2001.

[35] Laurent Perron and Vincent Furnon. OR-Tools. https://developers.google.com/optimization/, 2019.

[36] Gerhard Reinelt. TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

[37] Bernard L Welch. The generalization of student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[38] Christian Bauckhage, Cesar Ojeda, Rafet Sifa, and Stefan Wrobel. Adiabatic Quantum Computing for Kernel k= 2 Means Clustering. In *LWDA*, pages 21–32, 2018.

[39] Eldan Cohen, Arik Senderovich, and J Christopher Beck. An Ising framework for constrained clustering on special purpose hardware. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 130–147. Springer, 2020.

[40] Eldan Cohen, Avradip Mandal, Hayato Ushijima-Mwesigwa, and Arnab Roy. Ising-based consensus clustering on specialized hardware. In *International Symposium on Intelligent Data Analysis*, pages 106–118. Springer, 2020.

[41] Davis Arthur and Prasanna Date. Balanced k-Means Clustering on an Adiabatic Quantum Computer. *arXiv e-prints*, pages arXiv–2008, 2020.

[42] Prasanna Date, Davis Arthur, and Lauren Pusey-Nazzaro. Qubo formulations for training machine learning models. *Scientific Reports*, 11(1):1–10, 2021.

[43] Masayuki Ohzeki. Breaking limitation of quantum annealer in solving optimization problems under constraints. *Scientific reports*, 10(1):1–12, 2020.